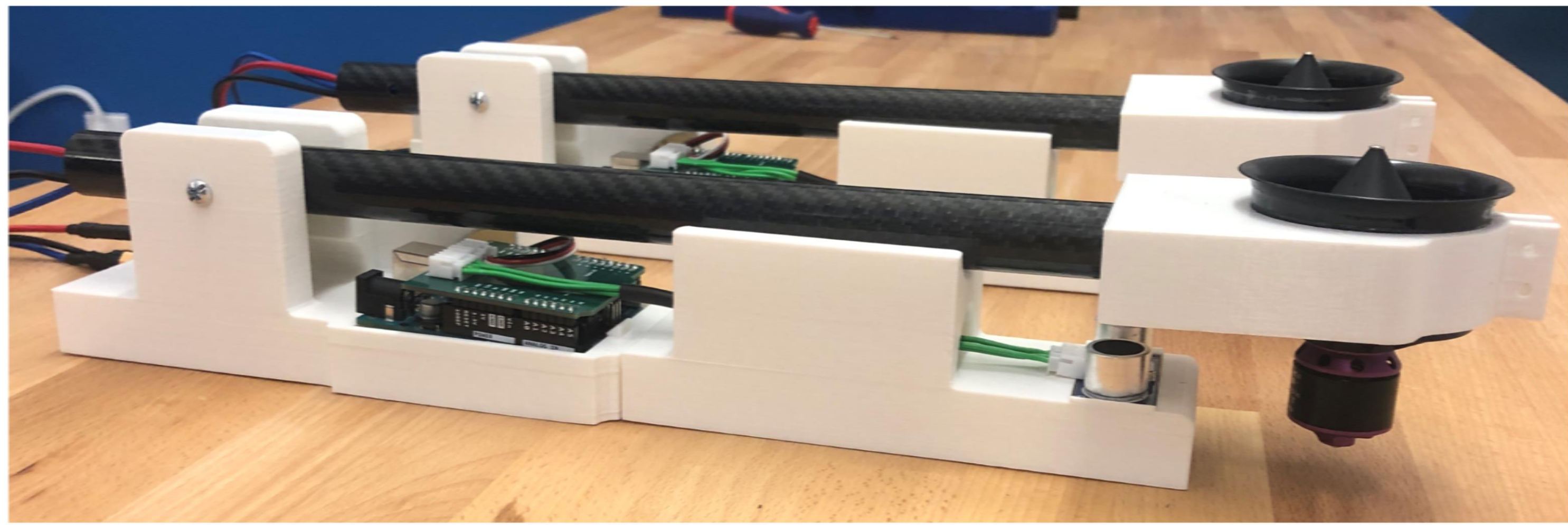


EasyGA/ GA Robot Project Overview

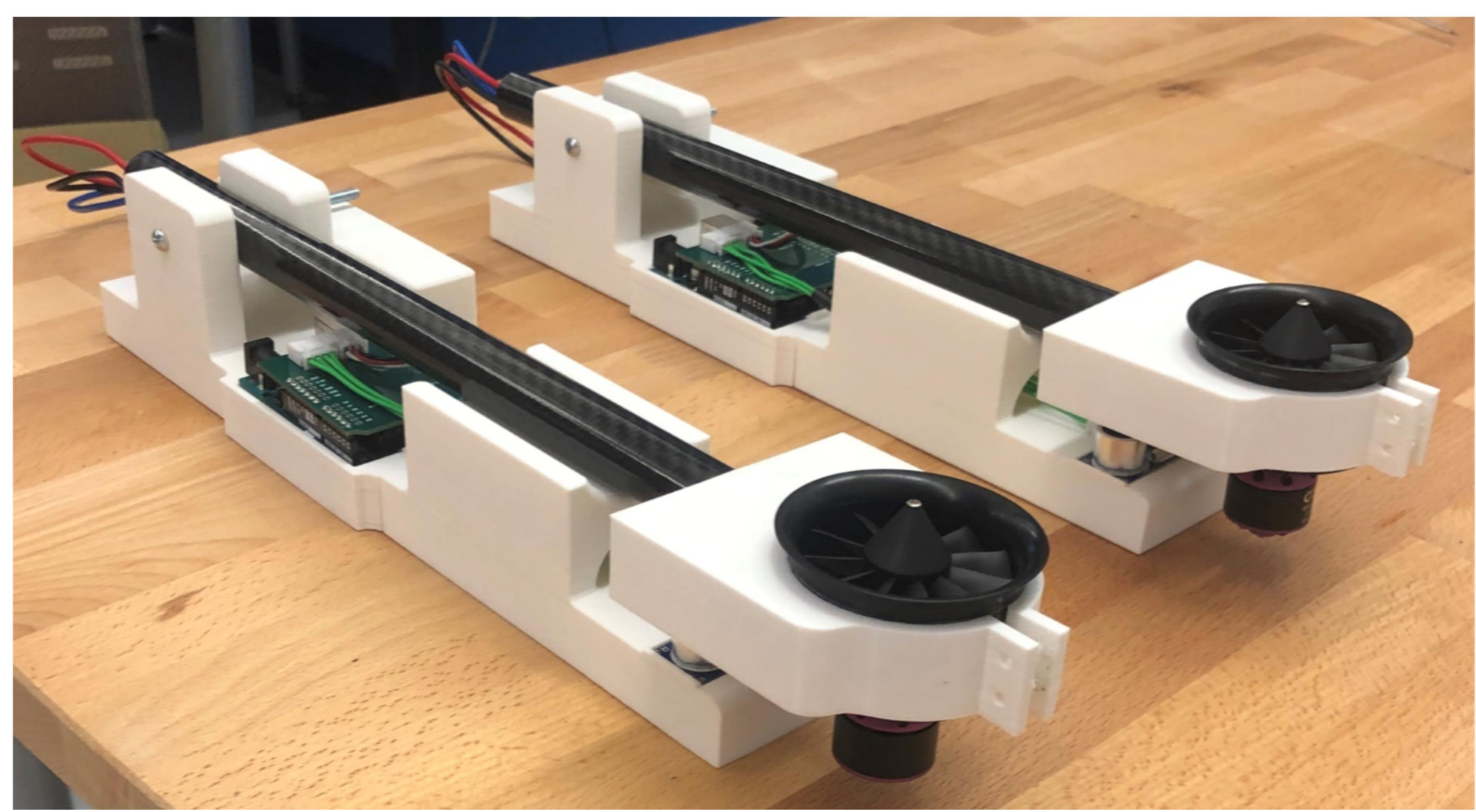
- Build open-source A.I software to apply in real time
- Build multiple A.I test platforms.

PHASE 1: GA ROBOT

- **Prior Research in Genetic Algorithm to Search the search space for the most optimal hover control.**



- The one DOF device is an evolving altitude controller. The device is made up of an EDF fan, the USRM, a carbon fiber tube, assorted wiring, and two 3D printed parts, the intermediary connector, and platform.

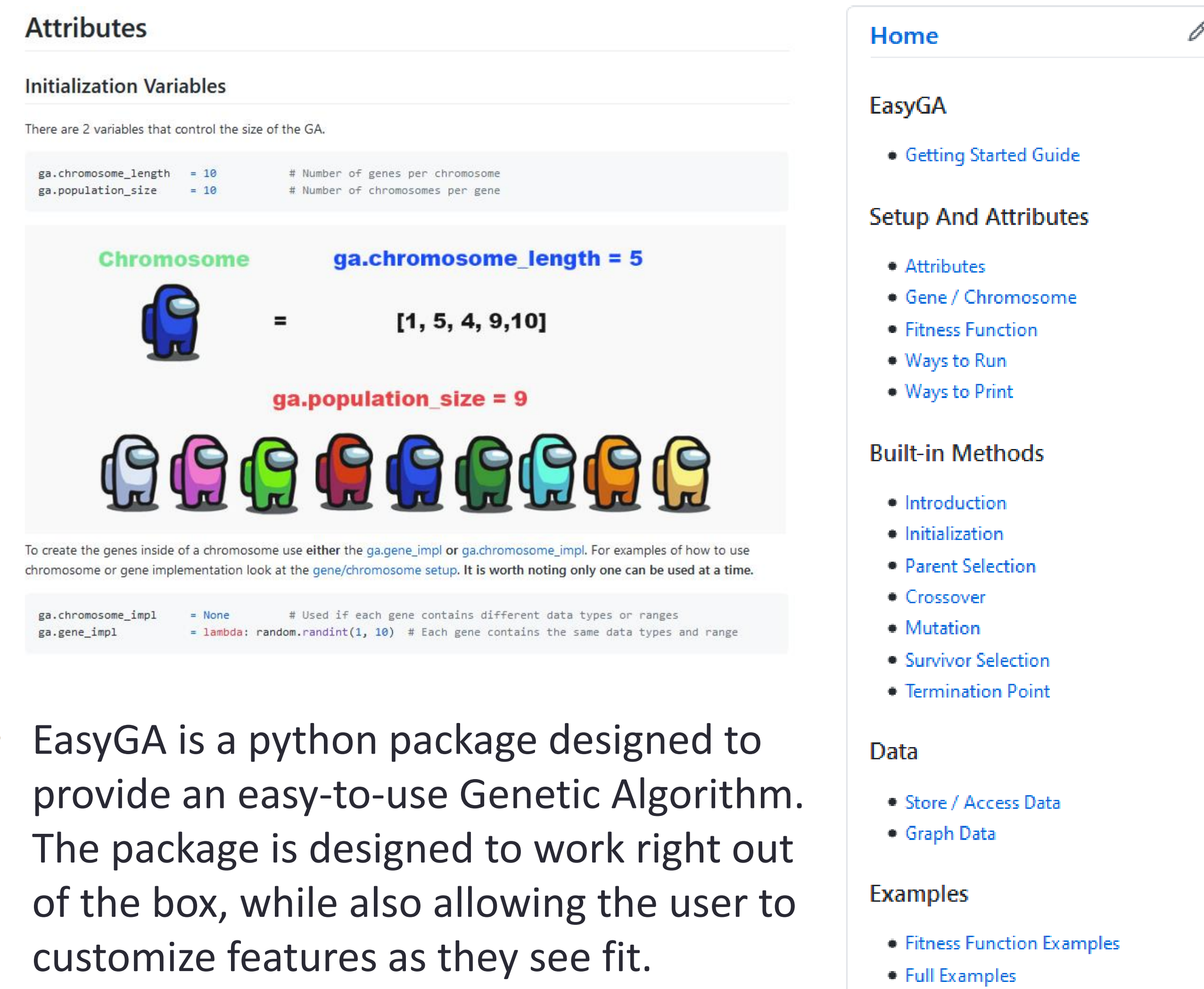


Abstract.

- The purpose of the present study was to design a flight control system with no pre-determined mathematical model, but instead using a genetic algorithm to maintain the optimal altitude. The study is done through a quantitative empirical research method. In the process of conducting the research, we found that programming a genetic algorithm was cumbersome for novice users to implement. Due to this, we created and released an open-source Python package called EasyGA.
- An initial population of 15 chromosomes and 100 generations were used during the trial. The throttle value of the device had an associated gene value of 1 second. When the trial of hundred generations was completed, machine learning was achieved. Please refer to the graph for greater understanding. Preliminary results showed that optimizing a one DOF device, in real-time, is possible without using a pre-determined mathematical model.

EASYGA SOFTWARE

- **Open-source software that makes genetic algorithms and neural networks easy and useable for anyone.**



Attributes

Initialization Variables

There are 2 variables that control the size of the GA.

ga.chromosome_length	= 10	# Number of genes per chromosome
ga.population_size	= 10	# Number of chromosomes per gene

Chromosome **ga.chromosome_length = 5**

ga.population_size = 9

To create the genes inside of a chromosome use either the `ga.gene_impl` or `ga.chromosome_impl`. For examples of how to use chromosome or gene implementation look at the [gene/chromosome setup](#). It is worth noting only one can be used at a time.

ga.chromosome_impl	= None	# Used if each gene contains different data types or ranges
ga.gene_impl	= lambda: random.randint(1, 10)	# Each gene contains the same data types and range

Home

EasyGA

- [Getting Started Guide](#)

Setup And Attributes

- [Attributes](#)
- [Gene / Chromosome](#)
- [Fitness Function](#)
- [Ways to Run](#)
- [Ways to Print](#)

Built-in Methods

- [Introduction](#)
- [Initialization](#)
- [Parent Selection](#)
- [Crossover](#)
- [Mutation](#)
- [Survivor Selection](#)
- [Termination Point](#)

Data

- [Store / Access Data](#)
- [Graph Data](#)

Examples

- [Fitness Function Examples](#)
- [Full Examples](#)

- EasyGA is a python package designed to provide an easy-to-use Genetic Algorithm. The package is designed to work right out of the box, while also allowing the user to customize features as they see fit.

Downloading Python

- EasyGA runs on Python 3.0+
- If you don't already have Python installed, you can install it here: <https://www.python.org/downloads/>

Pip Installing

- EasyGA can be easily installed using pip
- Pip comes pre-installed with all Python versions after 2.7
- But, if you don't already have pip installed, you can follow these instructions to install: <https://pip.pypa.io/en/stable/installing/>
- Once pip is installed, you can run the following line in the terminal to install EasyGA:

```
$ pip install EasyGA
```

Importing EasyGA

- Importing EasyGA into your .py file is as easy as adding the following line of code to the top of the file:

```
import EasyGA
```

EasyGA Example

```
import EasyGA

# Create the Genetic algorithm
ga = EasyGA.GA()

# Evolve the genetic algorithm
ga.evolve()

# Print your default genetic algorithm
ga.print_generation()
ga.print_population()
```

Output

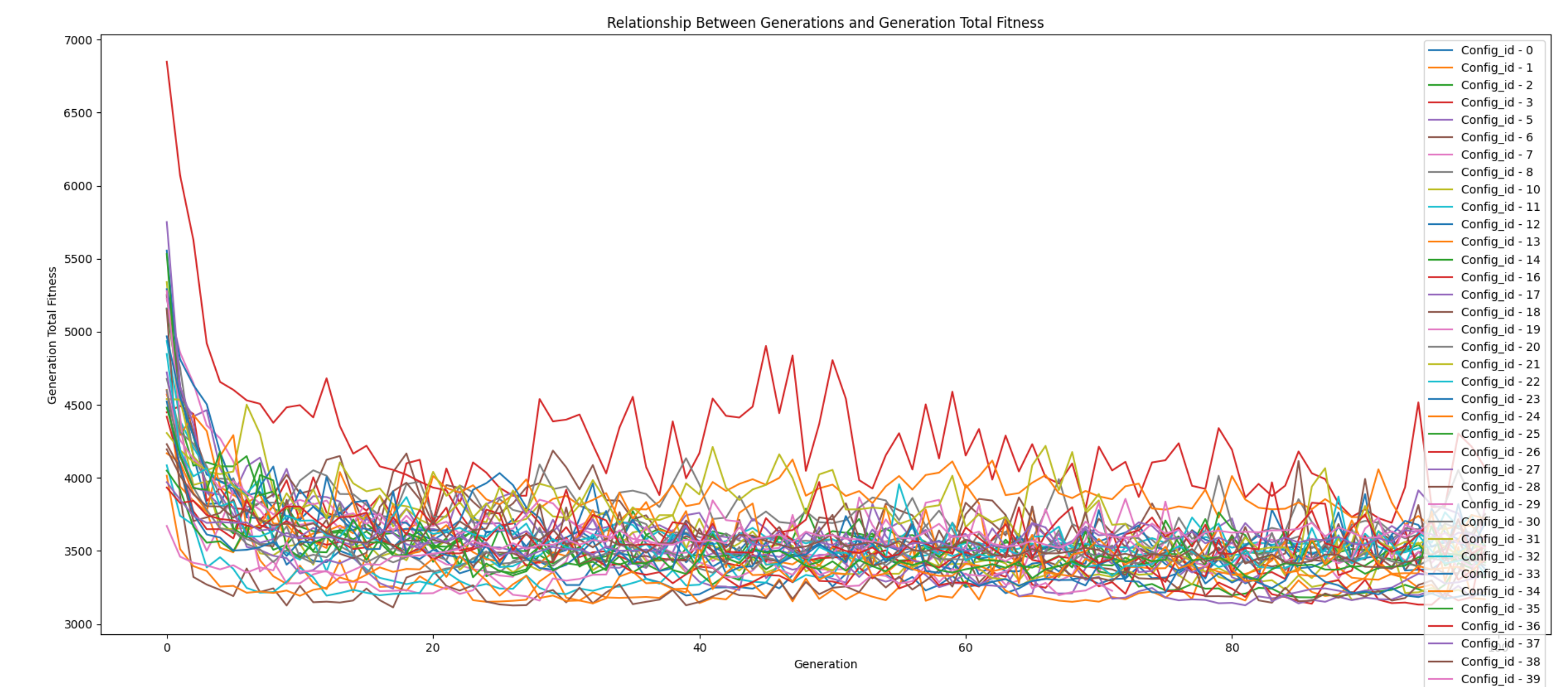
```
Current Generation: 15
Current population:
Chromosome - 0 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 1 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 2 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 3 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 4 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 5 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 6 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 7 [8][10][5][5][4][1][2][9][5][3] / Fitness = 3
Chromosome - 8 [8][10][5][5][4][1][2][9][5][3] / Fitness = 2
Chromosome - 9 [7][1][4][7][9][2][6][5][2][5] / Fitness = 2
```

PHASE 1 RESULTS:

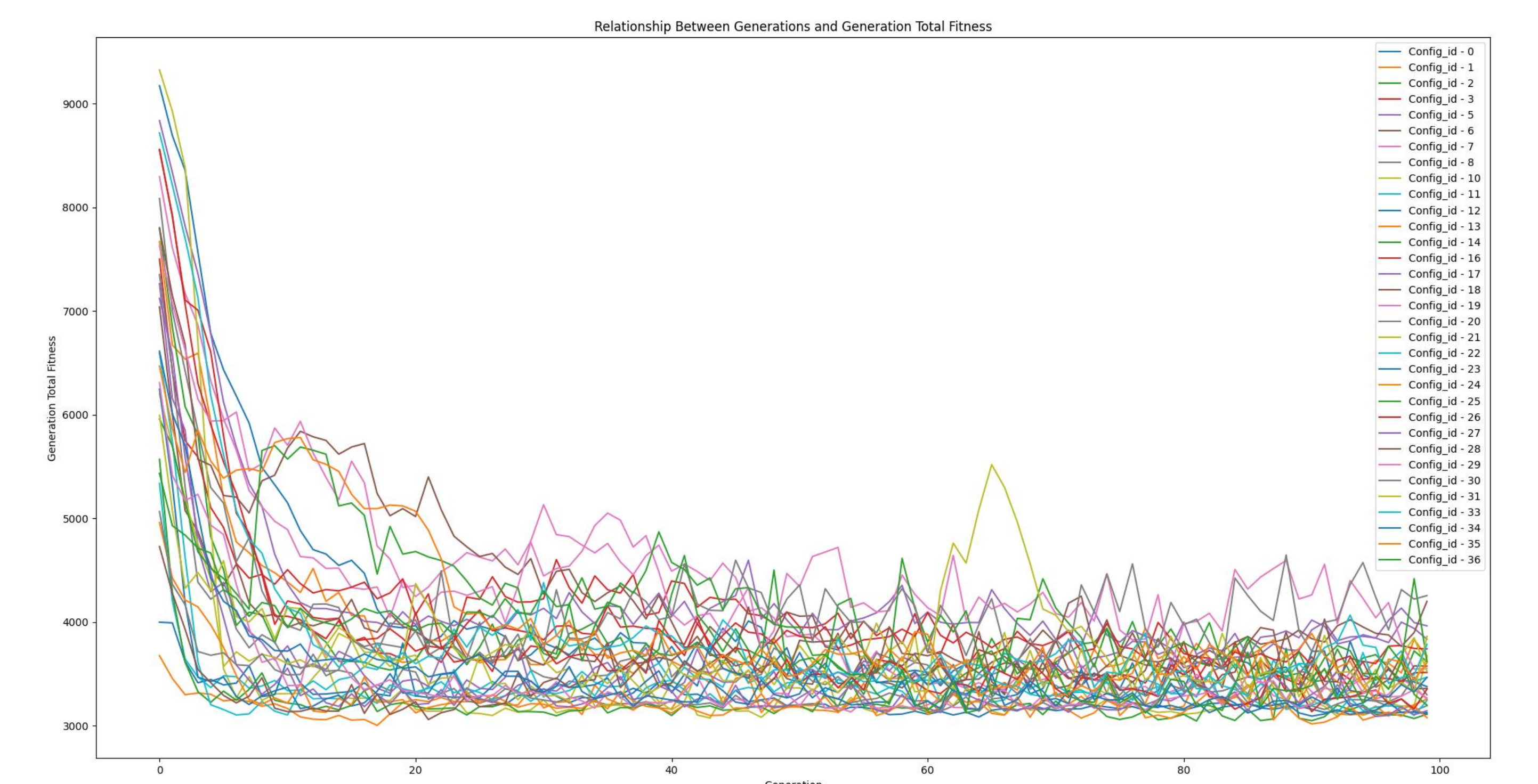
Preliminary Results:

- In the study, three robots(Kyle, Linda and Jeff) were made to test independently. Each robot used the exact some genetic algorithm hyperparameters and phases of ga to achieve its learning. These hyperparameters were: 15 chromosomes, 100 generations and chromosome length of 10 representing 1 section of flight each.

Robot Linda(35 Trials):



Robot Kyle(33 Trials):



Robot Jeff(30 Trials):

- Currently jeff is still being collected on because of multiple failed hardware and data corruption.

- The data may seem messy, but a clear pattern immerges within the robots finding a minimum around 200 which means it was not able to hit 6 cm in height for 2 out of 10 seconds or a finally score of 8/10.

- So easy, that your first genetic algorithm can be made in 5 lines of code. Seriously 5 lines of code.

- Multiple examples to use when your first getting started.

- No need to know every part of the algorithm. Just change what your need or know.